WHAT IS CLAIMED IS:

1. In a system for executing write and read data commands, the system having a buffer pool of blocks for temporarily storing write data to be sent to a peer device and read data received from the peer device, an apparatus for managing read and write data congestion in the buffer pool, the apparatus comprising:

a processor programmed for preventing an initiation of a new read or write data command until pending read and write data commands have been processed enough to free up sufficient blocks in the buffer pool to accommodate the data of the new read or write data command.

2. The apparatus as recited in claim 1, the processor further programmed for:

determining a number of blocks that will be required to store the read or write data for the new read or write data command;

determining a number of free blocks in the buffer pool; and

throttling the new read or write data command if the number of free blocks is insufficient to store the read or write data for the new read or write data command.

3. The apparatus as recited in claim 2, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the processor further programmed for determining the number of free blocks in the buffer pool by:

summing the number of free blocks in the receive list memory and the free list memory.

4. The apparatus as recited in claim 2, the processor further programmed for initiating the new read or write data command if the number of free blocks is sufficient to store the read or write data for the new read or write data command.

5. The apparatus as recited in claim 2, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the processor further programmed for determining the number of free blocks in the buffer pool by:

summing the number of free blocks in the receive list memory and the free list memory; and

adding the sum to a number of blocks estimated for storing incoming read data for any pending read data commands.

6. The apparatus as recited in claim 5, the processor further programmed for executing the new read or write data command if the number of free blocks is sufficient to store the read or write data for the new read or write data command.

7. The apparatus as recited in claim 2, the processor further programmed for:

storing the throttled new read or write data command and any subsequent new read or write data commands into a first-in-first-out (FIFO) read/write command request queue;

processing pending read data commands to completion to free up blocks in the buffer pool; and

executing a next read or write data command from the read/write command request queue if the number of free blocks becomes sufficient to store the read or write data for the next read or write data command.

8. A host bus adapter (HBA) comprising the apparatus of claim 1, the HBA for implementing upper layer protocols (ULPs).

9. The HBA of claim 8, further comprising an Internet Small Computer System Interface (iSCSI) controller circuit.

10. A host computer comprising the HBA of claim 9.

11. A storage area network (SAN) comprising the host computer of claim 10, wherein an iSCSI network is coupled to the iSCSI controller circuit and one or more storage devices are coupled to the iSCSI network.

12. A computer program for avoiding read and write data congestion in a system for executing write and read data commands and having a buffer pool of blocks for temporarily storing write data to be sent to a peer device and read data received from the peer device, the computer program being stored on a machine readable medium and executable to perform acts comprising:

preventing an initiation of a new read or write data command until pending read and write data commands have been processed enough to free up sufficient blocks in the buffer pool to accommodate the data of the new read or write data command.

13. The computer program as recited in claim 12, further executable to perform acts comprising:

determining a number of blocks that will be required to store the read or write data for the new read or write data command;

determining a number of free blocks in the buffer pool; and

throttling the new read or write data command if the number of free blocks is insufficient to store the read or write data for the new read or write data command.

14. The computer program as recited in claim 13, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the computer program further executable to perform acts comprising determining the number of free blocks in the buffer pool by:

summing the number of free blocks in the receive list memory and the free list memory.

15. The computer program as recited in claim 13, further executable to perform acts comprising initiating the new read or write data command if the number of free blocks is sufficient to store the read or write data for the new read or write data command.

16. The computer program as recited in claim 13, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the computer program further executable to perform acts comprising determining the number of free blocks in the buffer pool by:

summing the number of free blocks in the receive list memory and the free list memory; and

adding the sum to a number of blocks estimated for storing incoming read data for any pending read data commands.

17. The computer program as recited in claim 16, further executable to perform acts comprising executing the new read or write data command if the number of free blocks is sufficient to store the read or write data for the new read or write data command.

18. The computer program as recited in claim 13, further executable to perform acts comprising:

storing the throttled new read or write data command and any subsequent new read or write data commands into a first-in-first-out (FIFO) read/write command request queue;

processing pending read data commands to completion to free up blocks in the buffer pool; and

executing a next read or write data command from the read/write command request queue if the number of free blocks becomes sufficient to store the read or write data for the next read or write data command.

19. A host bus adapter (HBA) comprising the computer program of claim 12, the HBA for implementing upper layer protocols (ULPs).

20. The HBA of claim 19, further comprising an Internet Small Computer System Interface (iSCSI) controller circuit.

21. A host computer comprising the HBA of claim 20.

22. A storage area network (SAN) comprising the host computer of claim 21, wherein an iSCSI network is coupled to the iSCSI controller circuit and one or more storage devices are coupled to the iSCSI network.

23. A method for avoiding read and write data congestion in a system for executing write and read data commands and having a buffer pool of blocks for temporarily storing write data to be sent to a peer device and read data received from the peer device, the method comprising:

preventing an initiation of a new read or write data command until pending read and write data commands have been processed enough to free up sufficient blocks in the buffer pool to accommodate the data of the new read or write data command.

24. The method as recited in claim 23, further comprising:

determining a number of blocks that will be required to store the read or write data for the new read or write data command;

determining a number of free blocks in the buffer pool; and

throttling the new read or write data command if the number of free blocks is insufficient to store the read or write data for the new read or write data command.

25. The method as recited in claim 24, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the step of determining the number of free blocks in the buffer pool further comprising:

summing the number of free blocks in the receive list memory and the free list memory.

26. The method as recited in claim 24, further comprising initiating the new read or write data command if the number of free blocks is sufficient to store the read or write data for the new read or write data command.

27. The method as recited in claim 24, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the step of determining the number of free blocks in the buffer pool further comprising:

summing the number of free blocks in the receive list memory and the free list memory; and

adding the sum to a number of blocks estimated for storing incoming read data for any pending read data commands.

28. The method as recited in claim 27, further comprising executing the new read or write data command if the number of free blocks is sufficient to store the read or write data for the new read or write data command.

29. The method as recited in claim 24, further comprising:

storing the throttled new read or write data command and any subsequent new read or write data commands into a first-in-first-out (FIFO) read/write command request queue;

processing pending read data commands to completion to free up blocks in the buffer pool; and

executing a next read or write data command from the read/write command request queue if the number of free blocks becomes sufficient to store the read or write data for the next read or write data command.